

## A PLURALITY OF LOGICAL INTERFACES TO SHARED COPROCESSOR RESOURCE

Publication number: JP2002149424 (A)

Publication date: 2002-05-24

Inventor(s): DAVIS GORDON TAYLOR; MARKO C HEADS; RIWENS ROSE BOYED; RINALDI MARK ANTHONY

Applicant(s): IBM

Classification:

- international: G06F9/38; G06F9/46; G06F13/10; G06F13/14; G06F15/00; G06F15/16; G06F15/163; G06F15/76; G06F15/78; G06F15/80; H04L29/06; G06F9/38; G06F9/46; G06F13/10; G06F13/14; G06F15/00; G06F15/16; G06F15/76; H04L29/06; (IPC1-7): G06F9/46; G06F9/38

- European: G06F15/78F2

Application number: JP20010265792 20010903

Priority number(s): US20000656582 20000906

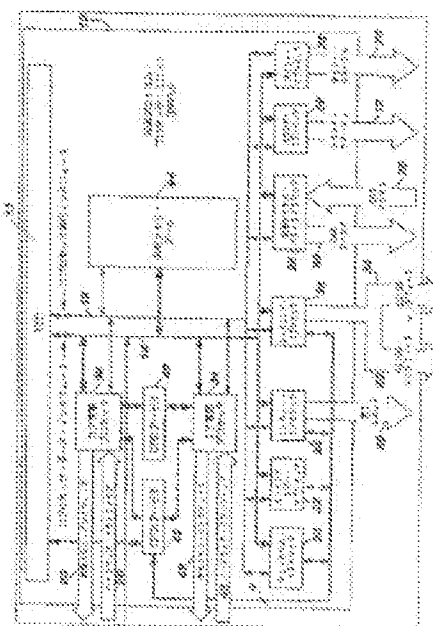
Also published as:

US6829697 (B1)  
TW581950 (B)  
SG100751 (A1)  
KR20020020186 (A)  
CN1342940 (A)

more >>

### Abstract of JP 2002149424 (A)

**PROBLEM TO BE SOLVED:** To increase the communicating efficiency of a protocol processor unit (PPU) and a coprocessor in a network processor. **SOLUTION:** An integrated processor composite body is provided with a plurality of protocol processor units (PPU). The respective units are provided with at least one or preferentially two individually functioning core language processors (CLP). The respective CLP are allowed to support dual threads through a logical coprocessor execution/data interface with a plurality of exclusive coprocessors to be used for the respective PPU. In response to an operation instruction, the PPU identifies an events whose waiting time is long and an event whose waiting time is short, and controls and switches the priority order of the execution of threads based on the identification. Also, in response to the operation instruction, the conditional execution of the specific coprocessor operation is made available when the designated specific event is generated or not generated.



Data supplied from the esp@cenet database — Worldwide

(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号  
特開2002-149424  
(P2002-149424A)

(43) 公開日 平成14年5月24日 (2002.5.24)

(51) Int.Cl. <sup>7</sup>	識別記号	F I	データ <sup>*</sup> (参考)
G 0 6 F 9/46	3 6 0	C 0 6 F 9/46	3 6 0 B 5 B 0 1 3
			3 6 0 C 5 B 0 9 8
9/38	3 7 0	9/38	3 7 0 C

審査請求 有 請求項の数29 O L (全 21 頁)

(21) 出願番号 特願2001-265792 (P2001-265792)

(22) 出願日 平成13年9月3日 (2001.9.3)

(31) 優先権主張番号 0 9 / 6 5 6 5 8 2

(32) 優先日 平成12年9月6日 (2000.9.6)

(33) 優先権主張国 米国 (US)

(71) 出願人 390009531

インターナショナル・ビジネス・マシーンズ・コーポレーション  
INTERNATIONAL BUSIN  
ESS MACHINES CORPO  
RATION  
アメリカ合衆国10504、ニューヨーク州  
アーモンク (番地なし)

(74) 代理人 100086243

弁理士 坂口 博 (外2名)

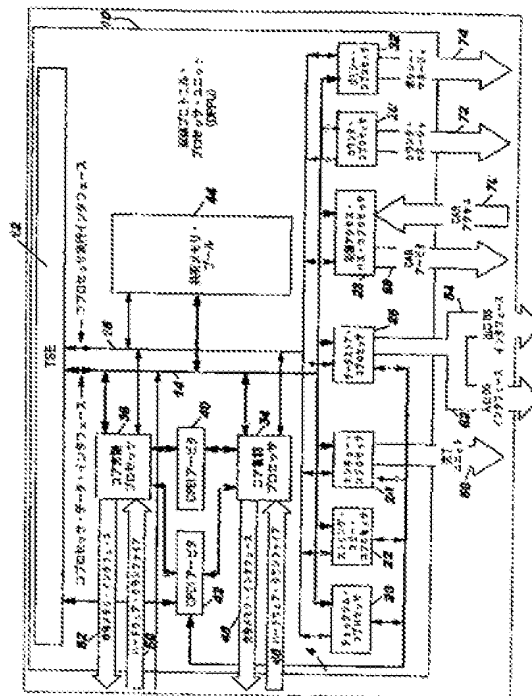
最終頁に続く

(54) 【発明の名称】 共有コプロセッサ・リソースに対する複数の論理インタフェース

(57) 【要約】 (修正有)

【課題】 ネットワーク・プロセッサにおいて、プロトコル・プロセッサ・ユニット (P P U) とコプロセッサとの通信効率を上げる。

【解決手段】 組み込みプロセッサ複合体は複数のプロトコル・プロセッサ・ユニット (P P U) を含む。各ユニットに少なくとも1つの、好適には2つの個別に機能するコア言語プロセッサ (C L P) が含まれる。各C L Pは、各P P Uに用いられる複数の専用コプロセッサとの論理コプロセッサ実行/データ・インタフェースを通してデュアル・スレッドをサポートする。操作命令により、P P Uが待ち時間の長いイベントと短いイベントを識別し、この識別をもとにスレッド実行の優先順位を制御し切り替える。また操作命令により、指定された特定のイベントの発生時または非発生時、特定のコプロセッサ・オペレーションの条件付き実行が可能になる。



## 【特許請求の範囲】

【請求項1】 ネットワーク・プロセッサのプログラミング機能を制御する組み込みプロセッサ複合体のオペレーションであって、該プロセッサ複合体は、複数のプロトコル・プロセッサ・ユニット（PPU）を含み、各PPUは少なくとも1つのコア言語プロセッサ（CLP）を含み、各CLPは少なくとも2つのコード・スレッドを持ち、各PPUはPPUの特定のタスクを実行する上で有用な複数のコプロセッサ及び複数の論理コプロセッサ・インタフェースを利用し、各CLPと該コプロセッサ間のアクセスを実現する、オペレーション。

【請求項2】 前記コプロセッサは、各CLPの複数のコード・スレッドをサポートする専用コプロセッサを含む、請求項1記載のオペレーション。

【請求項3】 前記コプロセッサは、ツリー検索コプロセッサ、チェックサム・コプロセッサ、ストリングコピー・コプロセッサ、エンキュー・コプロセッサ、データストア・コプロセッサ、CABコプロセッサ、カウンタ・コプロセッサ、及びポリシ・コプロセッサを含むグループから選択される、請求項1記載のオペレーション。

【請求項4】 複数のスレッド間の優先順位を確認するためコプロセッサ実行インタフェース・アービタを含む、請求項3記載のオペレーション。

【請求項5】 データ・スレッド間の優先順位を確認するためコプロセッサ・データ・インタフェース・アービタを含む、請求項3記載のオペレーション。

【請求項6】 各スレッドと少なくとも1つのコプロセッサ間にFIFOバッファを含む、請求項3記載のオペレーション。

【請求項7】 前記FIFOバッファは各スレッドと前記カウンタ・コプロセッサの間にある、請求項6記載のオペレーション。

【請求項8】 前記FIFOバッファは各スレッドと前記ポリシ・コプロセッサの間にある、請求項6記載のオペレーション。

【請求項9】 ネットワーク・プロセッサのプログラミング機能を制御する組み込みプロセッサ複合体を含むネットワーク処理システムであって、該複合体は複数のプロトコル・プロセッサ・ユニット（PPU）を含み、各PPUは、それぞれ少なくとも2つのコード・スレッドを持つ少なくとも1つのコア言語プロセッサ（CLP）と、前記システムの特定のタスクを実行する複数のコプロセッサ及び該コプロセッサのリソースにアクセスし各CLPと共有する複数のコプロセッサ・インタフェースと、を含む、システム。

【請求項10】 前記コプロセッサ・インタフェースは、各CLPのコード・スレッドをサポートすることにのみ用いられる、請求項9記載のネットワーク処理システム。

【請求項11】 前記コプロセッサは、ツリー検索コプロセッサ、チェックサム・コプロセッサ、ストリングコピー・コプロセッサ、エンキュー・コプロセッサ、データストア・コプロセッサ、CABコプロセッサ、カウンタ・コプロセッサ、及びポリシ・コプロセッサを含むグループから選択される、請求項10記載のネットワーク処理システム。

【請求項12】 各スレッドと前記コプロセッサのうち少なくとも1つの間にFIFOバッファを含む、請求項10記載のネットワーク処理システム。

【請求項13】 前記FIFOバッファは各スレッドと前記カウンタ・コプロセッサの間にある、請求項12記載のネットワーク処理システム。

【請求項14】 前記FIFOバッファは各スレッドと前記ポリシ・コプロセッサの間にある、請求項12記載のネットワーク処理システム。

【請求項15】 前記CLPのスレッドにより実行される特定の操作命令を含み、該実行の結果、コプロセッサ・オペレーションを制御するコマンドが得られ、該コマンドは前記CLPとコプロセッサ間のインタフェースを流れる、請求項9記載のネットワーク処理システム。

【請求項16】 前記命令は、特定のコプロセッサ・オペレーションの条件付き実行を可能にするように働く、請求項15記載のネットワーク処理システム。

【請求項17】 前記命令により、前記システムが、特定のコプロセッサ・コマンドに応答してデータにアクセスするための予測応答時間に従って、待ち時間の長いイベントと待ち時間の短いイベントを識別し、アクティブなスレッドの実行が待ち時間の長いイベントにより中断したときに他のスレッドにフル制御を与えるか、またはアクティブなスレッドの実行が待ち時間の短いイベントにより中断したときに他のスレッドに一時的制御を与える、請求項15記載のネットワーク処理システム。

【請求項18】 複数のプロトコル・プロセッサ・ユニット（PPU）を含む組み込みプロセッサ複合体内の命令の実行を制御する方法であって、該プロトコル・プロセッサ・ユニットはそれぞれ少なくとも1つのコア言語プロセッサ（CLP）を含み、該CLPはそれぞれ少なくとも2つのコード・スレッドを持ち、該方法は、該PPUに対する特定のタスクを実行するため、各PPUによる複数のコプロセッサの使用、及び該コプロセッサと各CLP間のアクセスを提供する複数の論理コプロセッサ・インタフェースの使用を含む、方法。

【請求項19】 前記PPUの複数のコード・スレッドをサポートする専用コプロセッサの使用を含む、請求項18記載の方法。

【請求項20】 前記コプロセッサの1つ以上は、ツリー検索コプロセッサ、チェックサム・コプロセッサ、ストリングコピー・コプロセッサ、エンキュー・コプロセッサ、データストア・コプロセッサ、CABコプロセッサ

サ、カウンタ・コプロセッサ、及びポリシ・コプロセッサを含むグループから選択される。請求項19記載の方法。

【請求項21】実行スレッド間の優先順位を確認するためコプロセッサ実行インタフェース・アービタが用いられる。請求項20記載の方法。

【請求項22】データ・スレッド間の優先順位を確認するためコプロセッサ・データ・インタフェース・アービタが用いられる。請求項20記載の方法。

【請求項23】各スレッドと前記コプロセッサのうち少なくとも1つの間にFIFOバッファを提供するステップを含む。請求項20記載の方法。

【請求項24】前記FIFOバッファは各スレッドと前記カウンタ・コプロセッサの間にある。請求項23記載の方法。

【請求項25】前記FIFOバッファは各スレッドと前記ポリシ・コプロセッサの間にある。請求項23記載の方法。

【請求項26】前記CLPにより実行される特定の操作命令を提供するステップを含み、該実行の結果、コプロセッサ・オペレーションを制御するコマンドが得られ、該コマンドは前記CLPとコプロセッサの間のインタフェースを流れる。請求項18記載の方法。

【請求項27】前記操作命令により特定のコプロセッサ・オペレーションの条件付き実行が可能になる。請求項26記載の方法。

【請求項28】前記実行は直接的または間接的である。請求項27記載の方法。

【請求項29】前記システムが、特定のコプロセッサ・コマンドに関する予測応答時間に従って、待ち時間の長いイベントと待ち時間の短いイベントを識別し、アクティブなスレッドの実行が待ち時間の長いイベントにより中断したときに他のスレッドにフル制御を与えるか、またはアクティブなスレッドの実行が待ち時間の短いイベントにより中断したときに他のスレッドに一時的制御を与える、命令を提供するステップを含む。請求項18記載の方法。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】本発明は、ネットワーク・プロセッサ・システムに関し、特に1つ以上のプロトコル・プロセッサ・ユニット（PPU）を含む組み込みプロセッサ複合体に関する。プロセッサ装置内でPPUと複数のコプロセッサを相互接続するインタフェースを通して、それらの間でデータや命令を転送するため、PPUとともに複数のコプロセッサが用いられる。

【0002】

【従来の技術】ネットワーク・プロセッサのプログラミング機能を実現し、これを制御するためプロトコル・プロセッサ・ユニットが用いられていることは周知の通り

である。同様に、コンピュータ・システム処理複合体アーキテクチャを設計する際に、PPUとともにコプロセッサが用いられることも一般的になっている。リアルタイム処理を必要とする処理イベントの遅れは、システム性能に直接影響を与える問題である。PPUによりタスクを実行するのではなく、特定のコプロセッサにタスクを割当てることによって、コンピュータ・システムの効率と性能を上げることができる。PPUがコプロセッサと効率よく通信することが重要である。この通信を改良し続けることが求められる。

【0003】

【発明が解決しようとする課題】本発明の目的は、1つ以上のコア言語プロセッサ（CLP）を含み、各CLPが複数のスレッドを持ち、論理コプロセッサ・インタフェースを通して特別タスク・コプロセッサに指示を与えるプロトコル・プロセッサ・ユニット（PPU）を使用することである。

【0004】本発明の他の目的は、共有コプロセッサ・リソースにアクセスするため（プログラマから見て）複数の論理コプロセッサ・インタフェースを使用することである。コプロセッサ・リソースは、PPU内の複数の処理スレッドにより共有されることがあり、複数のPPU間で1つのコプロセッサ・リソースが共有されることもある。

【0005】本発明の他の目的は、PPUとそのコプロセッサのインタフェース側で有効になる特定の操作に関する。この操作の1つは、コプロセッサ命令を条件付きで実行する機能である。これは特にカウンタ・コプロセッサで有効であるが、一般には他のコプロセッサにも適用できる。コプロセッサ・インタフェースは、特定のコプロセッサ・コマンドに関する予測応答時間に従って、待ち時間の長いイベントと短いイベントを識別する機能を持つ。この識別により、スレッドを実行する優先順位が制御される。

【0006】本発明の他の目的は、これまでのコプロセッサ・インタフェースに比べてフレキシビリティと効率が向上したコプロセッサ・インタフェースを提供することである。

【0007】

【課題を解決するための手段】以上の目的は、他の目的を含めて、以下に述べるようにして達成される。

【0008】ネットワーク・プロセッサのプログラミング機能を制御する組み込みプロセッサ複合体の動作について説明する。プロセッサ複合体は、複数のプロトコル・プロセッサ・ユニット（PPU）を含み、PPUはそれぞれ1つ以上のコア言語プロセッサ（CLP）を含む。CLPはそれぞれ複数のコード・スレッドを持つ。各PPUが、PPUに対して特定のタスクを実行する上で有用な複数のコプロセッサを利用する。複合体は、複数の論理コプロセッサ・インタフェースを使用し、CLP

Pとの共有コプロセッサ・リソースにアクセスする。CLPにより特定の動作命令が実行され、これによりコプロセッサにコマンドが送られる。これらの命令の1態様は、特定のコプロセッサ命令の条件付き実行を可能にすることである。命令は、特定のコプロセッサ・コマンドに関する予測応答時間に従って、待ち時間の長いイベントと短いイベントを識別することができる。これにより複合体は、処理されている待ち時間の長さタイプに応じて、制御をスレッドからスレッドへ切り替えることができる。

【0009】

【発明の実施の形態】本発明については、ネットワーク・プロセッサのプログラミング機能を提供し制御する埋め込みプロセッサ複合体の文脈で説明する。複合体の実施形態は通常、ハードウェア・アクセラレータと連携して高速パターン検索、データ操作、内部チップ管理機能、フレーム解析、及びデータのプリフェッチをサポートする8つのメイン処理ユニットまたはプロトコル・プロセッサ・ユニット（PPU）を含む。各PPUはそれぞれ構造コンポーネントを含み、構造コンポーネントは2つのCLPと、少なくとも1つ、好適には数個の専用／共有コプロセッサ・ユニット、及びメイン処理ユニットと各コプロセッサ・ユニットのインタフェースを含む。

【0010】各コプロセッサ・ユニットは、特定のネットワーク・タスクを実行することができる。メイン処理ユニットは、記憶プログラムの一連の命令を実行する。コプロセッサ・ユニットはそれぞれ、該メイン処理ユニットを担当し、メイン処理ユニットの制御下で特定のタスクを効率よく実行するようにされる。メイン処理ユニットと各コプロセッサ・ユニットのインタフェースにより、次の機能のうち1つ以上が有効になる。各コプロセッサ・ユニットの構成、各コプロセッサ・ユニットにより完了する特定のタスクの起動、各コプロセッサ・ユニットに関するステータス情報へのアクセス、及び各コプロセッサ・ユニットにより完了する特定のタスクに関する結果を返す手段の提供である。メイン処理ユニットとコプロセッサ・ユニットはそれぞれ1つ以上の専用レジスタを含む。インタフェースは、専用レジスタを該メイン処理ユニットとコプロセッサ・ユニットから共通アドレス・マップにマップすることができる。

【0011】各PPUはそれぞれ1つ以上のコア言語プロセッサ（CLP）及び数個の専用コプロセッサを含む。PPUに数個のCLPが含まれるとき、コプロセッサはCLP間で共有される。

【0012】各CLPはそれぞれ1つの演算論理ユニット（ALU）を含み、2つのコード・スレッドをサポートする（PPU毎に合計4つのスレッド）。CLPとコプロセッサは、スカラ・レジスタとアレイ・レジスタを含む専用レジスタの個別コピーを格納する。特定のコプ

ロセッサは、CLPからコプロセッサのアレイ・レジスタやスカラ・レジスタへのコプロセッサ・コマンドの転送を調整するFIFOバッファを含む。CLPは一度に1つのスレッド・コマンドのみ実行する。プログラマから見ると、各スレッドはそれぞれ自体のコプロセッサまたはコプロセッサ・セットに見える。コプロセッサのほとんどは、専用機能を実行し、相互に並行して、またCLPとともに動作することができる。

【0013】PPUの制御記憶は、通常、内部メモリと外部メモリの両方により与えられる。例えば即時アクセスには内部SRAMを、高速アクセスには外部ZBT SRAMを、大容量が求められる場合にはDDR SDRAMを使用できる。

【0014】図1に、ツリー検索エンジン12、チェックサム・コプロセッサ20、ストリング・コピー・コプロセッサ22、エンキュー・コプロセッサ24、データストア・コプロセッサ26、制御アクセス・バス・コプロセッサ28、カウンタ・コプロセッサ30、及びポリシ・コプロセッサ32を含む数個のコプロセッサとのデータ・インタフェース14と実行インタフェース16を維持するプロトコル・プロセッサ・ユニット（PPU）10を示す。

【0015】PPUは、コア言語プロセッサ（CLP）のペア34及び36を含む。各CLPに命令フェッチ・デコード／実行ユニット、複数の専用レジスタ、汎用レジスタ、及び2つのスレッドが含まれる。コプロセッサ実行インタフェース（CPEI）アービタ40は、2つのCLPとコプロセッサ間で命令を調停する。コプロセッサ・データ・インタフェース（CPDI）アービタ42は、コプロセッサとCLP34及び36間で通信の優先順位を確認する。CLPの命令は全て命令メモリ（図1には示していない）に保存される。

【0016】CLP#1 34は、バス46を通してハードウェア・クラシファイア（classifier）からの通信を受信する。ハードウェア・クラシファイアは、ディスパッチャからの刺激を与え、新しいバケットの処理を開始する。CLPは、命令メモリからのフェッチ命令をバス48を通して送り、新しいバケットを処理する。同様にCLP36は、バス50を通してハードウェア・クラシファイアから分類結果を受信し、バス52を通して命令メモリ・インタフェースに要求を送る。命令メモリ・インタフェースとハードウェア・クラシファイアはPPUの外部にあり、図1には示していない。

【0017】エンキュー・コプロセッサ24はバス60を通して外部完了ユニット（図示せず）に命令を送る。データストア・コプロセッサ26は、データをバス62を通して入口データストア・インタフェースに、またはバス64を通して出口データストア・インタフェースに送る。制御データのフローは、バス68を通して外部制御アクセス・バス・アービタ（図示せず）により調停さ

れる。アクセスの調整は、書込まれているか読取られているデータがバス70を通して流れている間にCABアービタ68上で行われる。データ・アクセスはバス70を通して入力または出力される。カウンタ・コプロセッサ30とポリシ・コプロセッサ32はそれぞれ、カウンタ・マネージャとポリシ・マネージャへのアクセスを、それぞれバス72及び74を通して提供する。

【0018】図2に、本発明に従った2つのCLPの他の詳細を示す。各CLPはそれぞれ汎用レジスタ80と専用レジスタ82を含む。これら専用レジスタは複数のスカラ・レジスタ84とアレイ・レジスタ86を含む。また命令フェッチ／デコード／実行の機能88も含む。

【0019】処理複合体は次のコンポーネントを含む。複数のプロトコル・プロセッサ・ユニット。好適な実施例では、サーバが8つのプロトコル・プロセッサ・ユニット（PPU）を使用する。各PPUに、複数（図では7つ）のコプロセッサを共有する1つ以上のCLPが含まれる。PPUは、フレームを転送し、テーブルを更新し、ネットワーク処理ユニットを維持するためのコードを実行する。

【0020】CLPは、共通命令メモリに保存されたコードを並行して実行する。各CLPはコアと3ステージのパイプライン、16のGPR（汎用レジスタ）、及びALU（演算論理ユニット）を含む。コプロセッサは操作を互いに並行して、またCLPと並行して実行することができる。コプロセッサは、CLPとインタフェースを取る際、基本CLP命令とレジスタ・モデルを拡張する。各コプロセッサのコマンドは新しいアセンブラ・ニューモニックとして現れ、コプロセッサのレジスタは、CLPプログラマから見て新しいスカラ・レジスタ、及びアレイ・レジスタとして現れる。アレイ・レジスタの一部は共有メモリ・プール44に位置する。コプロセッサはCLPと非同期に実行可能である。これによりCLPは、コプロセッサがコマンドを実行している間に命令の処理を続けることができる。CLPは、待機命令によりコプロセッサのコマンド実行が完了するまで待機する。

【0021】命令メモリ56は通常、8つの組み込みRAMで構成される。これらは、初期化時にロードされ、フレームを転送し、システムを管理するプログラム命令を格納する。命令メモリは、データ転送タスク、管理トラフィック、及び制御タスクのため16Kの命令を保持する。

【0022】ディスパッチャ・ユニット。これはスレッドの使用を管理し、新しいフレームをフェッチしアイドル・スレッドにディスパッチする。データ・フレームは、次に使用できるPPUにディスパッチされる。これによりアップ／ダウン・ディスパッチャ・キュー（up-GDQ、up-GCQ、dn-GRO/1、dn-GBO/1、及びdn-GCQ）からフレーム・アドレスのエンキューが解除される。エンキュー解除の後、ディ

スパッチャ・ユニットはアップ／ダウン・データストア（DS）からフレーム・ヘッダの一部を読み取り、これを共有メモリ・プール44に保存する。CLPがアイドルになるとすぐ、ディスパッチャ・ユニットがコード命令アドレス（CIA）等の対応する制御情報をバス46または50を介してCLPに渡す。ディスパッチャは別のバス58を使用してヘッダ情報を共有メモリ・プール44に送る。ディスパッチャはまたタイマと割込みを処理するため、それらの機能の作業を利用できるスレッドにディスパッチする。

【0023】ツリー検索メモリ（TSM）アービタ110。内部と外部に複数の共有メモリ位置があり、各CLPから利用できる。このメモリは共有されるので、アービタはメモリ・アクセス制御に用いられる。TSMはコードによって直接アクセスでき、例えばTSMにルーティング・テーブルを保存するため使用できる。またTSMは、ツリー検索時にTSE12によりアクセスされる。

【0024】完了ユニット（CU）。完了ユニットは2つの機能を実行する。第1に、CLPとUp/Dn EDS（エンキュー、エンキュー解除、及びアイランドのスケジュール（Schedule Island）のインタフェースを取る。EDSはエンキュー操作を実行し、これによりフレーム・アドレスがFCBPageと呼ばれるパラメータとともに転送キュー、破棄キュー、またはディスパッチャ・キューにエンキューされる。ターゲットが転送キューのとき、ハードウェアにより構成されるフロー制御機構によって、フレームを転送キューにエンキューするか、破棄キューにエンキューするか確認される。第2に、完了ユニットはフレーム・シーケンスを保証する。同じフローに属するフレームは複数のスレッドによって処理される可能性があるため、それらのフレームがUp/Dn転送キューに正しい順序でエンキューされるような予防措置が必要である。完了ユニットは、フレーム・ディスパッチ時にハードウェア・クラシファイア54により生成されるラベルを使用する。

【0025】ハードウェア・クラシファイア。これはディスパッチャ・ユニットとPPUの間のデータ・バスに置かれる。分類を行い、宛先スレッドに情報を提供する。Upフレームの場合、ハードウェア・クラシファイア・アシストがフレーム・フォーマットの周知のケースについて分類を行う。分類結果は、フレーム・ディスパッチの間、CIA（コード命令アドレス）及び4つのGPR（汎用レジスタ）の内容の形でCLPに渡される。Dnフレームの場合、ハードウェア・クラシファイア・アシストが、フレーム・ヘッダに応じてCIAを確認する。ハードウェア・クラシファイア・アシストは、Up及びDn両方のフレーム・ディスパッチでは、フレーム・シーケンスを維持するため完了ユニットにより用いられるラベルを生成する。

【0026】Up/Dnデータストア・インタフェースとアービタ。各スレッドからデータストア・コプロセッサ26を通してUp/Dnデータストアにアクセスすることができる。"他のデータ"を読取るときは読取りアクセスができ、データプールの内容をデータストアに書戻すときは書込みアクセスができる。複数のスレッドがあり、Upデータストアに一度にアクセスできるのは1スレッドのみ、Dnデータストアに一度にアクセスできるのも1スレッドのみなので、データストア毎に1つのアービタが必要である。

【0027】制御アクセス・バス(CAB)アービタとWEBWatchインタフェース。CABアービタはCABへのアクセスをスレッド間で調停する。スレッドは全て、CABコプロセッサ28を通してCABにアクセスすることができる。これによりプロセッサ・ユニットにある全てのメモリ及びレジスタの機能にアクセスできる。またどのスレッドも、全ての構成領域を変更または読取ることができる。CABは、プロセッサ・ユニットのメモリ・マップと考えることができる。

【0028】CABWatchインタフェースは、3つのチップI/Oを使用して、チップ外部からCAB全体へのアクセスを提供する。

【0029】デバッグ、割込み、シングル・ステップ制御。CABにより、GCHスレッドまたはCABWatchが各スレッドを制御することができる。例えばGFHスレッドまたはCABWatchがCABを使用して、シングル・ステップ実行モードで選択されたスレッドを実行することができる。

【0030】コア言語プロセッサ(CLP)：ネットワーク・サーバは、そのアーキテクチャで数種類のCLPを利用する。CLPの種類は、それぞれ特定の機能を実行するようにプログラムされる。

【0031】GDHは、汎用データ・ハンドラで、主にフレームの転送に用いられる。通常、GDHはそれぞれ専用制御プロセッサを持つ。制御プロセッサはそれぞれ、本発明に従って複数のコプロセッサを使用する。システムに必要なCLPの数は一般に、パフォーマンス評価により求められる。アーキテクチャと構造は完全にスケラブルであり、シリコン領域によってのみ制限される。CLPの数が増えると、シリコン領域に含まれるアービタと命令メモリが大きくなる。

【0032】管理セル・ハンドラ(GCH)のハードウェアはGDHと同じであるが、管理フレームはGCHによってのみ処理できる。GCHがデータ・フレームも処理するようになっている場合はWeb(CLP対応レジスタ)上でプログラミングできる(その場合GDHの役割を担う)。GCHは、ツリーの挿入や削除を行うため、GDHハードウェア・アシストにはないハードウェアを含む。GCHは、管理セル関連コードの実行、エンジンのようなチップ、ツリー管理関連コードの実行、

及びCPや他のGCHとの制御情報の交換に用いられる。そのような実行タスクがない場合、GCHはフレーム転送関連コードを実行し、その場合はGDHと全く同じように動作する。

【0033】汎用プロセッサ・ハンドラ(GPH)。このプロセッサは、Power PCに接続されるハードウェア・メールボックスにアクセスする。RIT1/2にはPower PCはないので、GPHはGDHと全く同じように動作する。

【0034】汎用ツリー・ハンドラ(GTH)には、ツリーの挿入、ツリーの削除、及びロープの管理を行うため、GDH及びGCHハードウェア・アシストにはないハードウェアがある。GTHは、GPQにツリー管理コマンドを含むフレームがないときデータ・フレームを処理する。

【0035】図2を参照する。CLP#1-34は、命令フェッチ/デコード/実行ユニット88、汎用レジスタ80、及びスカラ・レジスタ84とアレイ・レジスタ86が含まれる専用レジスタ82を含む。CLP#2-36も同種のコンポーネントを含む。

【0036】CLP34は、2つの命令を除いて、その実行ユニット102内で命令を完全に実行する。2つの例外は、図4の直接/間接コプロセッサ実行命令417である。これら2つの命令は、接続されたコプロセッサのうち1つでコマンド処理を開始する。コプロセッサはコマンドを互いに並行して実行でき、またCLP内の命令処理と並行して実行できる。CLP命令にコプロセッサがかかわるときは、コプロセッサ識別子と呼ばれ、操作のため選択されたコプロセッサを示す0乃至15の範囲内の4ビットの数が指定される。

【0037】共有メモリ・プール：

4Kバイトの共有メモリ・プール44は、コプロセッサの少なくとも一部に対するアレイ・レジスタを保持する。PPUで動作する全てのスレッドにより用いられる。各スレッドにより1Kバイトが用いられ、次の領域に分けられる。FCBpage(エンキュー・コプロセッサのアレイ・レジスタと見なされる)、データ・フェッチ、スクラッチ・メモリ領域(CLPのアレイ・レジスタと見なされる)、及びシステム領域である。プールはスレッド数に応じて、等しいセグメントに分けられていると見なすことができる。各セグメントでは、アドレス・スペースがCLP及びアレイ・レジスタを必要とする各種コプロセッサのアレイ・レジスタに分けられる。プールのアドレス・ラインのうち2つは、どのCLPがアクティブか、どのスレッドがアクティブかに応じて駆動される。

【0038】PPUCoproセッサ：コプロセッサはそれぞれ専用ハードウェア・アシスト・エンジンであり、コアに組み込まれた場合は、大量の直列化コードを必要とするような機能を実行する。コプロセッサはCLPと並

列に動作し、IPヘッダの変更、フロー制御アルゴリズムに用いられるフロー情報の維持、CABを介した内部レジスタへのアクセス、フロー制御及び管理情報ブロック(MIB)のカウンタの維持(標準とプロプライエタリ)、転送されるフレームのエンキュー等、データの移動に用いられる機能を提供する。プロセッサはそれぞれ、他に明記しない限り、PPUの各スレッドに対するスカラ・レジスタとアレイのセットを維持する。

【0039】再び図2を参照する。PPU10は2つのコア言語プロセッサ34、36と接続された数個のコプロセッサ12、20、22、24、26、28、30及び32を含む。これらのコプロセッサは、高速パターン検索、データ操作、内部チップ管理機能、フレーム解析、及びデータ・フェッチ等、特定のネットワーク処理タスクについてハードウェア・アクセラレーションを実現する。

【0040】以下、各種コプロセッサとその機能について説明する。

【0041】ツリー検索コプロセッサ：ツリー検索エンジン(TSE)コプロセッサ12には、コプロセッサ識別子2が割当てられる。TSEは、ツリー管理とアービタ110を介したツリー検索メモリへの直接アクセスに関するコマンドを持つ。LPM(可変長の一致を必要とする最長プレフィックス一致パターン)、FM(正確な一致のある固定サイズ・パターン)、及びSMT(範囲またはビット・マスク・セットを定義するパターンを伴うソフトウェア管理ツリー)の検索を行うアルゴリズムを持ち、フレームの転送、及び変更情報を取得する。コプロセッサ識別子1が割当てられるデータストア・コプロセッサ26は、フレーム・データの収集、変更、またはネットワーク・プロセッサのフレーム・データ・メモリ112への導入に用いられる。本発明に有用なツリー検索のアーキテクチャや動作の詳細については、米国特許出願ドケット番号RAL9990139、同RAL9990140、及びRAL9990141を参照されたい。

【0042】チェックサム・コプロセッサ：従来のチェックサム・コプロセッサ20は、インターネット・チェックサムを計算するため提供されるアルゴリズムを使用してチェックサムを計算し検証する。その際、ハーフワード・データに対してチェックサム操作を実行し、ハーフワード・チェックサム結果を返す。次のコマンドを使用できる。

- ・チェックサム生成、及び
- ・チェックサム確認

【0043】コマンドの結果は累算スカラ・レジスタとステイク・スカラ・レジスタに置かれる。累算スカラ・レジスタは、チェックサム計算の結果を格納し、ステイク・スカラ・レジスタは、チェックサムに含まれる最後のハーフワードに続くバイト位置を格納する。チェック

サム・コプロセッサのデータは共有メモリ・プールに置かれる。

【0044】コプロセッサに対するコマンドは次のオプションを含む。

1) IPヘッダ：IPヘッダが指示されたとき、レイヤ3ヘッダの開始位置(つまりステイク)が渡される。ハードウェアが、ヘッダ長フィールドからIPヘッダの長さを確認し、この値を長さスカラ・レジスタにロードする。チェックサムを生成する際、現在のチェックサムを格納したハーフワードの代わりに0の値が用いられる。

2) データ・ブロック：共有メモリ・プールにあるデータは、データのブロックとして処理され、チェックサムを生成または確認することができる。共有メモリ・プールの開始位置及び長さが渡される。データのブロックを確認する際、チェックサムは累算スカラ・レジスタに置かれる。データのブロックをチェックする際、チェックサムは累算レジスタに置かれている。

【0045】エンキュー・コプロセッサ：エンキュー・コプロセッサ24は2つの機能を提供する。

1) コードにより、ワーキングFCBPageと呼ばれ、アップ/ダウンFCBページを作成するため用いられる256ビット・レジスタ、FCB(フレーム制御ブロック)ページを作成することができる。レジスタは、フレームをEDS(エンキュー、デキュー/スケジューリング)アップまたはEDSダウンでエンキューするため必要な全てのパラメータを格納する。レジスタに格納されるパラメータの例として、アップのFCBアドレス、ターゲット・ポート番号、フレーム変更情報、及び次ループID等がある。

2) CLPと完了ユニット(CU)のインタフェースを提供する。CUはCLPから独立して動作するが、CLPプロセッサ毎にReady\_FCBPageというレジスタを格納する。レジスタは、エンキューの後、CUにコピーされ、その後、エンキューはエンキュー・プロセッサにより引き継がれる。その際、CLPが解放されて次のフレームが処理される。EQはそのレディ・ビット(?)を設定する。ただし、CUのReady\_FCBPageが空でない場合、EQは、EQレジスタが空になるまでCLPからCUへの転送をブロックし、その後転送を可能にする。エンキュー・コプロセッサは、スレッドと完了ユニットの間のインタフェース及び共有メモリ・プールに維持されるFCBPageの使用を管理する。各スレッドに3つのFCBPage位置があり、フレームに関するエンキュー情報をそこに維持することができる。ページのうち2つは、連続したエンキュー間で2つのページをスワップすることによって、完了ユニット・インタフェースに対するパフォーマンスを改良するため用いられる。スレッドに対して書かれるアセンブリ言語コードは、ハードウェアによって管理されるのでこれら2つのページを区別しない。3番目のページは、



コードにより新しいフレームを作成できるようにするためスレッドにより用いられる。この例として、学習のための管理トラフィック (guided traffic for learning) の作成がある。これはGTHスレッドにより実行されるように再エンキューされる。

【0046】CLPスレッドがエンキュー・コマンドを発行すると、FCBPageは使用中と指示される。他の位置が使用できる場合は、エンキュー・コプロセッサからの応答を待たずに新しいフレームがスレッドにディスパッチされる。完了ユニットは、エンキュー・コプロセッサを通して共有メモリ・プールからFCBPageをフェッチし、これをEDS (エンキュー・コマンドにより示される入口または出口) に提供する。これが起こるとFCBPageはフリーと指示される。両方のFCBPageが使用中と指示された場合、第3のフレームは起動できない。

【0047】エンキュー・コプロセッサでは次のコマンドがサポートされる。

- ・エンキュー入口 (ENQUP) は、完了ユニットを介して入口フロー制御/スケジューラにエンキューする。
- ・エンキュー出口 (ENQDN) は、完了ユニットを介して出口フロー制御/スケジューラにエンキューする。
- ・エンキュー・クリア (ENQCLR) は、現在のFCBPageをクリアする (全フィールドを0に設定する)。

【0048】データストア・コプロセッサ: このコプロセッサ26は次の機能を実行する。

- 1) アップ・データストア及びダウン・データストアとのインタフェースを取る。
- 2) タイマ・イベントのディスパッチまたは割込みのとき構成情報を受信する。
- 3) フレームのチェックサムを計算する。

このコプロセッサは通常、320バイトのデータ・バッファ及びそれぞれ128ビットのワード8個のメモリを含む。

【0049】フレーム・データはデータストア・コプロセッサを通してアクセスされ、メディアから受信されたフレームを格納する入口データストアと、パケット・ルーティング・スイッチから受信され再アセンブルされたフレームを格納する出口データストアとのインタフェースが取られる。また、タイマ・イベントのディスパッチや割込みのとき構成情報も受信される。

【0050】データストア・コプロセッサは、共有メモリ・プールで定義されたアレイを使用する。アレイはデータプールであり、8つのクォードワード及び2つのスクラッチ・アレイを保持でき、スクラッチ・アレイの1つは8つのクォードワード、もう1つは4つのクォードワードを保持する。データストア・コプロセッサには、入口、出口のデータストアとの間のアレイ内容の読取り、書き込みの制御に用いられる別のスカラ・レジスタが維持され

る。データストア・コプロセッサによりサポートされるスレッド毎に、アレイとスカラ・レジスタの1セットが定義されている。

【0051】これら共有メモリ・プールのアレイは、データストア・コプロセッサの作業領域になる。データストアを直接読取りまたはそこに直接書き込む代わりに、大量のフレーム・データがデータストアからこれら共有メモリ・プールのアレイに読取られるか、または大量のデータがこれらアレイからデータストアに書き込まれる。転送単位はクォードワードであり、クォードワードは16バイトとして定義される。

【0052】データストア・コプロセッサには次のコマンドを使用できる。

- 1) 出口データストア書き込み (WRDNDS) : CLPにより出口データストアへの書き込みができる。書き込みはクォードワード単位の倍数でのみ発生する。データはデータストア・コプロセッサのアレイのいずれか (データプールまたはスクラッチ・アレイ) から取られる。
- 2) 出口データストア読取り (RDDNDS) : CLPにより出口データストアからデータを読取り、データストア・コプロセッサのアレイの1つに入れることができる。読取りは、出口データストアに対して、クォードワード単位の倍数でのみ発生する。
- 3) 入口データストア書き込み (WRUPDS) : CLPにより入口データストアへデータを書き込むことができる。読取りは、入口データストアに対して、クォードワード単位の倍数でのみ発生する。
- 4) 入口データストア読取り (RDUPDS) : CLPにより入口データストアからデータを読取ることができる (クォードワード単位の倍数でのみ)。
- 5) 出口データストアからの他のフレーム・データの読取り (RDMOREDN) : 出口データストアからのハードウェア・アシスト読取り。RDMOREDNは、最後の読取りが停止したところからフレームの読取りを続け、データをデータプールに置く。データはデータプールに移されるので、ハードウェアは、読取られているフレームの現在位置を管理し、次のツイン・バッファの位置を確認するためツイン・バッファからリンク・ポイントをキャプチャする。このアドレスは、ツインが尽きて次のツインが読取られるまで、後続のRDMOREDN要求のためハードウェアにより用いられる。データプールの内容はツインの内容のマップなので、データプール内でフレーム・データがラップされる可能性がある。コードによりデータプール内のデータの位置が管理される。
- 6) 入口データストアからの他のフレーム・データの読取り (RDMOREUP) : 入口データストアからのハードウェア・アシスト読取り。RDMOREUPは、最後の読取りが停止したところからフレームの読取りを続け、データをデータプールに置く。データはデータプー

ルに移されるので、ハードウェアは、読取られているフレームの現在位置を管理し、フレームの次のデータ・バッファの位置を確認するためバッファ制御ブロック領域に維持されたリンクをキャプチャする。このアドレスは、データ・バッファが尽きて次のバッファが読取られるまで、後続のRDMOREUP要求のためハードウェアにより用いられる。コードによりデータプール内のフレームのデータの位置が管理される。

7) リース・ツイン・バッファ (LEASETWIN) : フリー・ツイン・バッファ (出口データストアで新しいデータを作成するとき用いられる) のアドレスを返す。

【0053】制御アクセス・バス (CAB) コプロセッサ : このコプロセッサ28では、ネットワーク・プロセッサが、ネットワーク・プロセッサ・システム全体で、選択されたレジスタを制御することができる。システム初期化等のために特定のレジスタを初期化でき、システム診断やメンテナンスのため特定のレジスタを読取ることができる。

【0054】コプロセッサは、組み込みプロセッサ複合体 (EPC) Webアービタとインタフェースを取る。アービタはCLPとWebウォッチ間の調停を行う。これによりCLPは全てWeb上で読取り、書込みができる。

【0055】CABコプロセッサは、CLPスレッドに関してCABアービタと制御アクセス・バスにインタフェースを与える。スレッドは、CABのアドレス、データ等、CABアクセスのオペランドをロードする必要がある。その場合、CABにアクセスするプロトコルは、CABインタフェース・コプロセッサにより処理される。CABインタフェース・コプロセッサは次のコマンドを提供する。

・CABアクセス調停 (WEBARB) : CABへのアクセスを取得するためスレッドにより用いられる。アクセスが取得されると、スレッドはCABを解放するまでCABの制御を維持する。

・CAB読取り/書込み (WEBACCESS) : CAB及びCABからアクセスできる接続されたレジスタとの間でデータを移動する。PPU内の送信元と宛先は汎用レジスタ (GPR) である。

・CAB優先使用 (WEBPREEMPT) : GFHスレッドによってのみ用いられ、これによりGFHは、1回の読取り/書込みアクセスについて、CABがすでに他のスレッドに与えられている場合でもCABの制御を取得する。

【0056】チェックサム、データストア、エンキュー、CABの各コプロセッサのアーキテクチャ、動作に関する他の詳細については、整理番号RAL9990083の米国特許出願、"String Copy (StrCopy) Coprocessor"を参照されたい。

【0057】ストリング・コピー・コプロセッサ22はCLPの機能を拡張し、データのブロックを移動する。データは共有メモリ・プール内でのみ移動する。次のコマンドが使用できる。

・ストリング・コピー (Strcopy) : このコマンドは、アレイ間でデータの複数のバイトを移動するため用いられる。コマンドは、ソース・データ・ブロックとシンク・データ・ブロックの開始バイト位置及び移動するバイト数を渡す。

【0058】カウンタ・コプロセッサ : カウンタ・コプロセッサ30は、全PPU間で共有できるカウンタ・マネージャ (図示せず) へのアクセスをバス72を通して提供する。コプロセッサは、全てのカウンタ・プログラムとインタフェースを取り、カウンタの更新を行う。スカラー・レジスタとコプロセッサ実行インタフェース16間のFIFOバッファ76で実装される。アレイ・レジスタとコプロセッサ・データ・インタフェース14間に第2FIFOバッファ78が置かれる。スレッドはそれぞれ自体のカウンタ・コプロセッサを持っているかのように動作する。このコプロセッサには外部 (PPUに対して) アドレス/データ・バスが用いられる。これにより、外部バスを通してカウンタ・コプロセッサを使用するため2つ以上のPPUが調停を行えるようにシステムを実装することができる。

【0059】スレッドは、カウンタ・コプロセッサを通してカウンタの更新を要求し、カウンタ・マネージャが操作を完了するのを待たずに処理を続けることができる。カウンタ・コプロセッサはその要求をカウンタ・マネージャに通知し、カウンタ・アクセス・コマンドのオペランドを処理のためカウンタ・マネージャに渡す。カウンタ・コプロセッサには、PPUで動作する4つのスレッドにより発行されたカウンタ・アクセス・コマンドを入れる8ディーブ・キューがある。カウンタ・コプロセッサは次のコマンドを提供する。

【0060】カウンタ・アクセス (CtrAccess) は、カウンタを増分するかまたはカウンタに値を追加する。コマンド・オペランドは、カウンタ識別子 (カウンタ・メモリの形)、インデックスとオフセット、増分もしくは追加コマンド、値フィールド、カウンタの読取りもしくは書込み、またはカウンタ値の読取りとクリアである。スレッドは、カウンタ・コプロセッサ・キューが一杯でなければ、コマンドの実行を待たない。

【0061】カウンタ・コプロセッサとその動作について詳しくは、整理番号RAL920000078US1の米国特許出願、"Coprocessor for Managing Large Counter Arrays"を参照されたい。

【0062】ポリシ・コプロセッサ : ポリシ・コプロセッサ32は、スレッドに関してポリシ・マネージャ (図示せず) とのインタフェース74を提供する。スカラー・レジスタとコプロセッサ実行インタフェース16間のF

FIFOバッファ76で実装される。アレイ・レジスタとコプロセッサ・データ・インタフェース14間には第2 FIFOバッファ78が置かれる。スレッドは、このインタフェースを通してフレームの“カラー”の更新を要求する。フレームのカラーは、ネットワーク・プロセッサの構成可能なフロー制御機構の一部として用いられ、この機構によりフレームに対する処理が決定される。スレッドはポリシ・マネージャが、ポリシ・コプロセッサを介して結果を返すまで待機する必要がある。ポリシ・マネージャは、このフレームがメンバーであるフローについてポリシ制御ブロックにアクセスする。オペランドには、ポリシ制御ブロック・アドレス、バケット長、フレームに現在割当てられているカラー等がある。返される結果はフレームの新しいカラーである。

【0063】CLP34、36はそれぞれ、コプロセッサ実行インタフェース16とコプロセッサ・データ・インタフェース14の2つのインタフェースを通してコプロセッサ12、20、22、24、26、28、30及び32に接続される。これらのインタフェースの機能については図4で詳しく説明する。

【0064】PPU内のコプロセッサはそれぞれ、4ビット・コプロセッサ識別子により識別される。各コプロセッサが最大256の専用レジスタをサポートする。コプロセッサ内の専用レジスタは、0乃至255の範囲の8ビット・レジスタ番号により識別される。コプロセッサ番号(CP#)とレジスタ番号の組み合わせにより、PPU内のレジスタが識別される。スカラ・レジスタとアレイ・レジスタの2種類の専用レジスタがある。

【0065】レジスタ番号0乃至239はスカラ・レジスタに予約されている。スカラ・レジスタは最小1ビット、最大32ビットである。スカラ・レジスタのビットには0乃至31までの番号が振られ、0は右端またはLSB (least significant bit)、31は左端またはMSB (most significant bit) である。32ビット未満の長さのスカラ・レジスタは右揃えされ、残りのビットは非実装と見なされる。CLPが32ビット未満の長さのスカラ・レジスタを読取る時、非実装ビットの値はハードウェアに依存する。非実装ビットへの書き込みは無効である。

【0066】レジスタ番号240乃至255はアレイ・レジスタに予約されている。アレイ・レジスタは最小2バイト、最大256バイトである。CLPはアレイ・レジスタを読取るか書き込み、共有メモリ・プール44内で一度に2バイト(ハーフワード)、一度に4バイト(ワード)、または一度に16バイト(クォードワード)パーティションをきる。

【0067】汎用レジスタの使用方法は周知の通りであり、ここでは一般的なことについて述べる。プログラマから見た汎用レジスタは2通りある。汎用レジスタは、0、2、4、...、14の集合から4ビット数で表され

る32ビット・ラベルで示されるように、32ビット・レジスタと見なすこともできる。この意味でプログラマは8個の32ビット汎用レジスタを扱う。またプログラマは汎用レジスタを、0、1、2、...、15の集合からの4ビット数として表される16ビット・ラベルに従って、16ビット・レジスタとして扱うこともできる。この意味でプログラマは16個の16ビット・レジスタを扱う。

【0068】各コプロセッサに、ビジー信号フィールドからの情報を格納するステータス・レジスタが含まれる。このレジスタは、所与のコプロセッサが利用できるか、またはビジー状態かどうかをプログラマに示す。コプロセッサ完了コード・レジスタが図4のOK/K、O、フィールド415からの情報を格納する。従って、プログラマは、所与のコプロセッサがビジーかまたは利用できるか知る必要がある場合、この情報をコプロセッサのステータス・レジスタから取得することができる。同様にコプロセッサ完了コード・レジスタが、コプロセッサ・タスクの完了についてプログラマに情報を提供する。

【0069】各CLPに次の16ビット・プログラム・レジスタが含まれる。プログラム・カウンタ・レジスタ、プログラム・ステータス・レジスタ、リンク・レジスタ、及びキー長レジスタである。タイムスタンプ・レジスタと乱数ジェネレータ・レジスタの2つの32ビット・レジスタも追加される。前記レジスタそれぞれにスカラ・レジスタ番号も与えられる。

【0070】プログラマから見た汎用レジスタは2通り考えられる。プログラマは汎用レジスタを32ビット・レジスタと見なすことができ、16ビット・レジスタと見なすこともできる。

【0071】アレイ・レジスタは、アレイ・レジスタ番号を通してプログラマに知られる。

【0072】図4は、コプロセッサ実行インタフェース16とコプロセッサ・データ・インタフェース14を通してCLP34をそのコプロセッサ401に接続するインタフェース信号を示す。個々のワイヤ接続数は、個々の割当てアイテムの矢印の横にある番号ラベルに示してある。この説明では、選択コプロセッサ20、22、...は、コプロセッサ識別子が、後述する操作に応じて411、420、または429に現れるコプロセッサ識別子に一致するコプロセッサを表す。

【0073】CLP34は、実行インタフェース16により、任意のコプロセッサ20、22、...、上でコマンドの実行を開始することができる。コプロセッサ番号411は、コマンドのターゲットとして16のコプロセッサのうち1つを選択する。CLTにより開始フィールド410が論理“1”になると、コプロセッサ番号411により示される選択コプロセッサ450が、6ビットOpフィールド412により指定されたコマンドの実行を

開始する。Op引数413は、44ビットのデータで、コプロセッサ450により処理されるようにコマンドとともに渡される。ビジー信号414は16ビット・フィールドで、各コプロセッサ401に1ビットであり、コプロセッサがコマンドを実行しているビジーか（ビット＝1）または、そのコプロセッサがコマンドを実行していない（ビット＝0）ことを示す。これら16ビットはスカラ・レジスタに保存され、レジスタのビット0はコプロセッサ0に、ビット1はコプロセッサ1に、以下同様に対応する。OK/K、O、フィールド415は16ビット・フィールドで、各コプロセッサ401に1ビットである。これは1ビット戻り値コードであり、コマンドに依存する。例えば、コプロセッサ401に与えられたコマンドが失敗に終わったか、コマンドが成功したかをCLP34に示すためこれを使用できる。この情報はCLPスカラ・レジスタ内に保存され、レジスタのビット0はコプロセッサ0に、ビット1はコプロセッサ1に、以下同様に対応する。直接/間接フィールド417は、コプロセッサ実行命令のどのフォーマットが実行されているかを選択コプロセッサ450に示す。直接/間接＝0のとき、直接フォーマットが実行されている。直接/間接＝1のときは間接フォーマットが実行されている。

【0074】コプロセッサ・データ・インタフェース14は3つの信号グループを含む。書き込みインタフェース419、420、421、422、423、424は、コプロセッサ内のスカラ・レジスタまたはアレイ・レジスタにデータを書き込むときに関係する。読取りインタフェース427、428、429、430、431、432、433は、コプロセッサ内のスカラ・レジスタ84またはアレイ・レジスタ86のいずれかの専用レジスタ82からデータを読取るときに関係する。第3のグループ425、426、427は、スカラ・レジスタまたはアレイ・レジスタの読取りと書き込みの両方で用いられる。読取りインタフェースと書き込みインタフェース両方に対する複製機能は、レジスタからレジスタへデータを移動するための同時読取り/書き込みをサポートするように働く。

【0075】書き込みインタフェースは、書き込みフィールド419を使用して、コプロセッサ番号420により示されるコプロセッサ450を選択する。書き込みフィールド419は、CLP34が選択コプロセッサにデータを書き込もうとするときに1に設定される。コプロセッサ・レジスタ識別子421は、CLP34が選択コプロセッサ内450に書き込もうとすることをレジスタに示す。コプロセッサ・レジスタ識別子421は8ビット・フィールドで、よって256のレジスタがサポートされる。0乃至239の範囲のコプロセッサ・レジスタ識別子はスカラ・レジスタへの書き込みを示す。240乃至255の範囲のコプロセッサ・レジスタ識別子はアレイ・レジ

スタへの書き込みを示す。アレイ・レジスタ書き込みの場合、オフセット・フィールド422は、アレイ・レジスタのデータ書き込み操作の開始点を示す。このフィールドは8ビットで、従ってアレイ内で256のアドレスをサポートする。データ出力フィールドは、コプロセッサ450に書き込まれるデータを入れる。これは128ビットの大きさで、従って最大128ビットの情報を一度に書き込むことができる。書き込み有効フィールド424は、コプロセッサ450がデータの受信をいつ終了したかをCLP34に示す。これによりCLP34は、コプロセッサ450がデータを取る間、一時停止し、データを有効な状態に保つことができる。

【0076】読取りインタフェース14は、書き込みインタフェース16と構造は似ているが、データはコプロセッサから読取られる。読取りフィールド428は書き込みフィールド419に対応し、選択コプロセッサ450で読取り操作がいつ実行されるかを示すためCLP34により用いられる。コプロセッサ番号識別子フィールド429は、どのコプロセッサ450が選択されているかを示す。レジスタ番号フィールド430、オフセット・フィールド431、及び読取り有効フィールド433は、書き込みインタフェースの421、422及び424に対応する。データ入力フィールド432は、コプロセッサ450からCLP34へのデータを入れる。

【0077】読取りまたは書き込みの操作は、3つの長さのうちいずれか1つをとる。16ビットが転送されることを示すハーフワード、32ビットが転送されることを示すワード、及び128ビットが転送されることを示すクォードワードである。読取りデータ432と書き込みデータ423は128ビット幅である。128ビット未満のデータ転送は右揃えされる。信号425及び426は、データ転送サイズを示す。16ビットの転送は425と426が両方とも0で示され、32ビットの転送は425と426がそれぞれ1と0で示され、128ビットの転送は425と426がそれぞれ0と1で示される。

【0078】修飾フィールド427は、データ読取りまたはデータ書き込みの操作で用いられる。コプロセッサはそれぞれ、コプロセッサのハードウェア・デザイナーにより定義された独自の方法でその意味を解釈する。これによりプログラマは、読取りまたは書き込みの操作のときにハードウェアに対して情報ビットを追加指定することができる。データストア・コプロセッサは、バケット・バッファのリンク・リストでバケット・バッファのリンク・フィールドを省略することができる。

【0079】コプロセッサでタスクを開始した後、CLPは、命令の実行を続けるか、またはコプロセッサでタスクが完了するまで実行を中断することができる。CLPが、コプロセッサ内のタスク実行と並行して命令の実行を続ける場合、後の時点で、メイン・プロセッサ・ユニットによるWAIT命令の実行のため、1つ以上のコ

プロセッサでのタスク実行が完了するまで他の命令の実行が中断する。WAIT命令は、形式によっては1つ以上のコプロセッサ内でタスクが完了するまでCLP上の実行を中断させる。その時点でCLPは命令の実行をWAIT命令に続く命令から再開する。他の形では、WAIT命令により、特定のコプロセッサ内でタスクが完了するまでCLPの実行が中断する。そのタスクが完了すると、CLPはコプロセッサからの1ビット戻りコードを、WAIT命令からの1ビットとともに調べ、WAIT命令に続く命令から命令の実行を再開するか、実行をプログラムにより指定された他の命令に分岐するか確認する。

【0080】コプロセッサ実行命令は、コプロセッサでのコマンド処理を真似るため、図1のコプロセッサ実行インタフェース16の“開始”信号を1に設定する。図5乃至8を参照する。コプロセッサ識別子520は、命令フィールド500から取得され、開始信号を介して選択コプロセッサを示す。6ビット・コプロセッサ・コマンドは命令フィールド501から取得され、どのコマンドの実行を開始するかを信号により選択コプロセッサに示す。開始信号がアクティブにされ1になると、選択コプロセッサはそのビジー信号をアクティブにし（1にする）、コマンドの実行を完了するまで1のままにしておく。コマンドの実行が完了すると、この信号を0にする（非アクティブ化）。CLPは継続して16ビットの信号を読み取り、それらをそのスカラ・レジスタに入れる。コマンドの完了後、選択コプロセッサはこのステータスをスカラ・レジスタに入れる。

【0081】再び図5乃至図8を参照する。命令の非同期実行フィールド502が0のとき、CLPはコマンドの完了を示すため、そのビジー信号を無効化する。そのとき、CLPは命令のフェッチと実行を再開する。命令の非同期実行フィールド502が1のとき、CLPは、ビジー信号の状態にかかわらず命令のフェッチと実行を続ける。

【0082】選択コプロセッサでコマンド処理が開始されると、CLPは44ビットの他のコマンド対応情報を信号によりコプロセッサに与える。この情報は、図5乃至図8に示すように命令フォーマットに応じて4つの方法のいずれかで引き出される。

【0083】図5のコプロセッサ実行間接フォーマットは、上位12ビット523のコマンド情報を命令フィールド504から取得する。下位32ビットのコマンド情報524は32ビット汎用レジスタ505から取得される。選択レジスタは、値{0, 2, 4, . . . , 14}に制限された4ビット命令フィールド503により求められる。こうしてそのレジスタから32ビット・レジスタが選択される。CLPは信号を1に設定し、これが命令の間接フォームであることを選択コプロセッサに示す。

【0084】実行命令の条件付きコプロセッサ実行間接

フォーマットを図6に示す。ここで命令は、満足した特定の条件をもとに図5と同じように実行される。条件を満足しない場合、命令は実行されない。命令は、CLPのALUコードをもとに実行される。条件付き実行には4ビットが使用され、その結果Opフィールドは2ビットに短縮される。従って考えられる64のコマンドのうち4つに対して条件付き実行が可能になる。他のコマンドは0と見なされる。よって長い待ち時間と短い待ち時間をもとにした条件付き実行を実現することができる。コプロセッサ命令の条件付き実行は特に、カウンタ・コプロセッサの動作と関連するときに有益である。

【0085】図7のコプロセッサ実行直接フォーマットは、下位16ビット527のコマンド情報を命令フィールド506から取得する。上位28ビット526のコマンド情報は0に設定される。CLPは信号を0に設定し、これが命令の直接形式であることを選択コプロセッサに示す。

【0086】条件付きコプロセッサ実行直接フォーマットを、図7と同じように実行されるように構成された形で図8に示す。図6と同様、条件付き実行では、4ビットが使用され、その結果Opフィールドは2ビットに短縮される。よって、考えられる64のコマンドのうち4つに対して条件付き実行が可能になる。

【0087】図9は、コプロセッサ待機命令のフォーマットを示す。CLPは、コプロセッサ・ステータス・レジスタで、命令フィールド600から取得された16ビット・マスクに対してビット単位でAND演算を行う。結果が0でない、つまり1つ以上のコプロセッサが現在まだコマンドを実行している場合、CLPは命令のフェッチと実行を中断する。ただし、前記AND演算の実行は、結果が0になるまで継続する。

【0088】図10は、コプロセッサ待機／分岐フォーマットを示す。コプロセッサ識別子フィールド601は、コプロセッサ・ステータスの特定のビットがテストされることを示す。例えばフィールド601に1があるとき、コプロセッサ・ステータス・レジスタのビット1がテストされる。識別子フィールド601に15があるとき、コプロセッサ・ステータスのビット15がテストされる。テストされるビットの値が1で、対応するコプロセッサがコマンドの実行をまだ完了していないことを示す場合、CLPは命令のフェッチと実行を中断する。ただし前記の演算はテスト・ビットの値が0、つまり対応するコプロセッサがコマンドの実行を完了するまで継続する。この時点で、命令のOKフィールド602の値と、コプロセッサ識別子601により選択された、スカラ・レジスタのコプロセッサ完了コードのビットの値に応じて、2つの処理のうちいずれかが発生する。CLPは、下表に応じて、次のシーケンシャル命令のフェッチと実行を再開するか、または分岐して、命令フィールド603により示される命令アドレスから命令のフェッチ

と実行を再開する。

602の値      選択コプロセッサ  
完了コード・ビットの値=0      完了コード・ビットの値=1

0      分岐  
1      次の命令

【0089】コプロセッサ・ユニットでタスクを開始するときの命令の実行については、米国特許出願第548109号、“Coprocessor Structure and Method for a Communications System Network Processor”を参照されたい。

【0090】本発明は、更に、各CLPの複数の命令実行スレッド（それぞれ、処理中の別々のパケットに関係する）それぞれの独立したプロセス及びデータ・アクセス時の待ち時間の処理に関する。実行スレッドはそれぞれ独立したプロセスであり、スレッドがコプロセッサ・ハードウェアにアクセスできるときに命令のシーケンスを実行する。ツリー検索コプロセッサは、パイプラインにされ、ツリー検索パイプラインで複数の実行スレッドがそれぞれ同時に、ただし異なるフェーズで（オーバーラップして）アクセスすることができる。本発明は、好適には、オーバーヘッドで複数の命令実行スレッドを採用し、実行をスレッドからスレッドに切り替える。スレッドはキューにされ、共有メモリに対するアクセスが高速に配信される。スレッドをキューにすることで、長い待ち時間のイベントに対して優先順位が最大のスレッドを可能な限り速く得ることができる。

【0091】前述のように、PPUはそれぞれ、実行スレッド毎に1つ、複数の命令プリフェッチ・バッファを含む。これらのプリフェッチ・バッファにより、アクティブな実行スレッドにより命令の帯域幅が十分利用されていないインターバル時、アイドル中の実行スレッドのため命令のプリフェッチが可能になる。これにより、制御が新しい実行スレッドに切り替わったとき、そのスレッドの命令プリフェッチ・バッファが一杯になりやすく、よって、実行に利用できる命令が足りないために新しいスレッドがすぐ中断する可能性がなくなる。こうして、命令メモリに対するアクセス優先順位は、現在実行中のスレッドが最大の優先順位になり、現在のスレッドが中断したときに制御を取る立場にある実行スレッドが第2の優先順位を与えられるように制御される。同様に、実行キューの下端にある実行スレッドには、命令フェッチ・アクセスで最後の優先順位を与えられる。

【0092】待ち時間の長いイベントのため（ツリー検索等）、アクティブなスレッドの実行が中断したとき、次のスレッドにフル制御が与えられるか、または待ち時間の短いイベント（ローカル・データ・ストレージでのコプロセッサの操作もしくは命令フェッチ待ち時間）のため実行が中断したときは、一時的制御が次のスレッドに与えられる。一時的制御が他のスレッドに与えられた場合、制御は、ブロックが解除されるとすぐに元のスレ

【表1】

選択コプロセッサ  
完了コード・ビットの値=1

次の命令  
分岐

ッドに返される。逆に他のスレッドにフル制御が与えられた場合、その他のスレッドは、ブロックされるまで制御を保つ。これにより待ち時間の短いイベントでのサイクルの無駄が避けられるが、1次実行スレッドが待ち時間の長いイベントに届くまでの時間は短くなる。他の場合、複数の実行スレッドがほぼ同じ時間に待ち時間の長いイベントに届く可能性があり、1つのスレッドのPPU実行を他のスレッドのツリー検索とオーバーラップするメリットは少なくなる。待ち時間をもとに制御を割り当てることについて詳しくは、整理番号RAL920000008の米国特許出願第542189号、“Network Processor with Multiple Instruction Threads”を参照されたい。この割当てとスレッド実行制御の詳細は次のようになる。

【0093】コンピュータの電源が初めて投入されたとき、各CLPスレッドは初期化状態にある。パケットがプロセッサにディスパッチされると、対応するスレッドがレディ状態に変わり、その時点で実行サイクル要求を開始する。

【0094】アービタが、アービタの論理関数をもとにブル式に従って実行サイクルをスレッドに与える。サイクルが与えられた場合、スレッドはレディ状態から実行状態に移る。実行状態のスレッドは、待ち時間イベントまたは処理されているパケットがエンキューされ、よってそのパケットに対するコードの作業が終了したことが示されたため実行が中断するまで要求を出し続ける。サイクルが与えられなくなると、これは他のスレッドが制御を取ったことを示す。これは、アービタがサイクルを与えない唯一の理由である。ただし、これら2つの状態のいずれか（レディまたは実行）で、スレッドは、パケット処理が終わりに達し、次のパケットがディスパッチのためキューに入るまで、新しい実行サイクルを要求し続け、待ち時間イベントで一時的に停止する。システムはそこで初期化状態に戻り、次のパケットを待機する。待機状態は、待ち時間の長いイベントまたは短いイベントを扱う。どのイベントが発生するかにかかわらず、プロセッサは中断し、アクティブなスレッドはデフォルトで待機状態になる。スレッドはそこで、待ち時間イベントが完了するまで実行サイクルの要求を停止する。

【0095】スレッドを初期化状態からレディ状態に移すのと同じディスパッチ操作により、スレッド番号がFIFOバッファに入り、第1パケットがディスパッチされるスレッドは、優先順位が最高のスレッドになる。後のディスパッチ操作では、他のスレッド番号がFIFOに送られる。FIFOの優先順位が最大のスレッド番号

は、待ち時間の長いイベントに出会うまでその位置にとどまり、その後、スレッドはFIFOの始めに戻り、最大優先順位から最小優先順位のスレッドに変わる。待ち時間の短いイベントによってスレッドがFIFOバッファで優先順位を失うことはない。

【0096】スレッドがバケットの処理を終了すると、バケットは、出力ポートに転送するためエンキューされ、スレッド番号がFIFOバッファから移される。

【0097】新しいバケットは、ハイレベル・コントローラ（図示せず）からディスパッチされる。このコントローラはスレッドとプロセッサを選択して各バケットを処理する。この決定により入力コマンドがFIFOバッファに送られる。また入力状態機械に送られ、初期化状態からレディ状態に移行することが状態機械に指示される。外部コントローラからのそのコマンドとともに、バケットのディスパッチ先であるスレッド番号もコントローラからFIFOに送る必要がある。

【0098】基本的に、実行を中断させるイベントは現在のプログラムの流れで短い割込みになるイベントと長い割込みになるイベントの2種類ある。短い割込みは、プログラムの流れが変わったため命令プリフェッチ・キューを再び埋める必要がある分岐命令により発生することがある。或いはまた、プログラムはコプロセッサがプロセッサのローカル・メモリでデータ関連タスクを実行するのを待機している間に中断することがある。この例は、チェックサム・コプロセッサが、変更されたヘッダ・フィールドで新しいチェックサムを計算する場合である。待ち時間が25プロセッサ・サイクル未満のとき、イベントは、短い割込みと見なされる。待ち時間の長いイベントは通常、25を超える待ち時間を伴い、通常は50プロセッサ・サイクル乃至100プロセッサ・サイクルを超える。これらは全体のパフォーマンスに大きな影響を与える。

【0099】待ち時間の長いイベント、短いイベントを確認する手段は他に多数ある。待ち時間の長さはプログラムが制御でき、その場合ハードウェアやその構成は確認の際の要素にはならない。一方、しきい値レジスタを25サイクルのしきい値で設定することもでき、その場合、操作に必要なサイクル数はハードウェアにより確認され、その確認をもとに自動的な判断が行われる。

【0100】コプロセッサ命令は、プロセッサが実行する命令の1タイプである。フィールドのビットの一部は、どのコプロセッサが対象かを示す。1ビットにより、特定の命令が待ち時間の長いイベントまたは短いイベントとして定義される。従って、プログラムは、同じアクセスを2つ定義することができる。1つは待ち時間の長いイベントとして、もう1つは待ち時間の短いイベントとして定義される。スレッド実行制御関数は、こうした待ち時間の長いイベントの影響を最小にするために設計されている。よって待ち時間の長いイベントによ

り、フル制御が別の実行スレッドに切り替わり、待ち時間の短いイベントにより一時的にのみ他のスレッドへの切り替えが起こる。

【0101】プロトコル・プロセッサ・ユニット（PPU）とコア言語プロセッサの詳細は、当業者には周知の通りであり、本発明の一部を構成しないが、それらは、変更或いは実装することでネットワーク・プロセッサ・システム全体のアーキテクチャの一部になっており、特定の機能コプロセッサやシステムの他のコンポーネントと連携する。本発明で有用な個々のコプロセッサのアーキテクチャやプログラミングを含めた詳細は、本発明の一部を構成するものと見なされない。

【0102】まとめとして、本発明の構成に関して以下の事項を開示する。

【0103】（1）ネットワーク・プロセッサのプログラミング機能を制御する組み込みプロセッサ複合体のオペレーションであって、該プロセッサ複合体は、複数のプロトコル・プロセッサ・ユニット（PPU）を含み、各PPUは少なくとも1つのコア言語プロセッサ（CLP）を含み、各CLPは少なくとも2つのコード・スレッドを持ち、各PPUはPPUの特定のタスクを実行する上で有用な複数のコプロセッサ及び複数の論理コプロセッサ・インタフェースを利用し、各CLPと該コプロセッサ間のアクセスを実現する。オペレーション。

（2）前記コプロセッサは、各CLPの複数のコード・スレッドをサポートする専用コプロセッサを含む、前記（1）記載のオペレーション。

（3）前記コプロセッサは、ツリー検索コプロセッサ、チェックサム・コプロセッサ、ストリングコピー・コプロセッサ、エンキュー・コプロセッサ、データストア・コプロセッサ、CABコプロセッサ、カウンタ・コプロセッサ、及びボリシ・コプロセッサを含むグループから選択される、前記（1）記載のオペレーション。

（4）複数のスレッド間の優先順位を確認するためコプロセッサ実行インタフェース・アービタを含む、前記（3）記載のオペレーション。

（5）データ・スレッド間の優先順位を確認するためコプロセッサ・データ・インタフェース・アービタを含む、前記（3）記載のオペレーション。

（6）各スレッドと少なくとも1つのコプロセッサ間にFIFOバッファを含む、前記（3）記載のオペレーション。

（7）前記FIFOバッファは各スレッドと前記カウンタ・コプロセッサの間にある、前記（6）記載のオペレーション。

（8）前記FIFOバッファは各スレッドと前記ボリシ・コプロセッサの間にある、前記（6）記載のオペレーション。

（9）ネットワーク・プロセッサのプログラミング機能を制御する組み込みプロセッサ複合体を含むネットワー



ク処理システムであって、該複合体は複数のプロトコル・プロセッサ・ユニット (P P U) を含み、各 P P U は、それぞれ少なくとも 2 つのコード・スレッドを持つ少なくとも 1 つのコア言語プロセッサ (C L P) と、前記システムの特定のタスクを実行する複数のコプロセッサ及び該コプロセッサのリソースにアクセスし各 C L P と共有する複数のコプロセッサ・インタフェースと、を含む、システム。

(10) 前記コプロセッサ・インタフェースは、各 C L P のコード・スレッドをサポートすることにのみ用いられる、前記 (9) 記載のネットワーク処理システム。

(11) 前記コプロセッサは、ツリー検索コプロセッサ、チェックサム・コプロセッサ、ストリングコピー・コプロセッサ、エンキュー・コプロセッサ、データストア・コプロセッサ、C A B コプロセッサ、カウンタ・コプロセッサ、及びポリシ・コプロセッサを含むグループから選択される、前記 (10) 記載のネットワーク処理システム。

(12) 各スレッドと前記コプロセッサのうち少なくとも 1 つの間に F I F O バッファを含む、前記 (10) 記載のネットワーク処理システム。

(13) 前記 F I F O バッファは各スレッドと前記カウンタ・コプロセッサの間にある、前記 (12) 記載のネットワーク処理システム。

(14) 前記 F I F O バッファは各スレッドと前記ポリシ・コプロセッサの間にある、前記 (12) 記載のネットワーク処理システム。

(15) 前記 C L P のスレッドにより実行される特定の操作命令を含み、該実行の結果、コプロセッサ・オペレーションを制御するコマンドが得られ、該コマンドは前記 C L P とコプロセッサ間のインタフェースを流れる、前記 (9) 記載のネットワーク処理システム。

(16) 前記命令は、特定の Coプロセッサ・オペレーションの条件付き実行を可能にするように働く、前記 (15) 記載のネットワーク処理システム。

(17) 前記命令により、前記システムが、特定の Coプロセッサ・コマンドにตอบสนองしてデータにアクセスするための予測応答時間に従って、待ち時間の長いイベントと待ち時間の短いイベントを識別し、アクティブなスレッドの実行が待ち時間の長いイベントにより中断したときに他のスレッドにフル制御を与えるか、またはアクティブなスレッドの実行が待ち時間の短いイベントにより中断したときに他のスレッドに一時的制御を与える、前記 (15) 記載のネットワーク処理システム。

(18) 複数のプロトコル・プロセッサ・ユニット (P P U) を含む組み込みプロセッサ複合体内の命令の実行を制御する方法であって、該プロトコル・プロセッサ・ユニットはそれぞれ少なくとも 1 つのコア言語プロセッサ (C L P) を含み、該 C L P はそれぞれ少なくとも 2 つのコード・スレッドを持ち、該方法は、該 P U に対す

る特定のタスクを実行するため、各 P P U による複数の Coプロセッサの使用、及び該 Coプロセッサと各 C L P 間のアクセスを提供する複数の論理 Coプロセッサ・インタフェースの使用を含む、方法。

(19) 前記 P P U の複数のコード・スレッドをサポートする専用 Coプロセッサの使用を含む、前記 (18) 記載の方法。

(20) 前記 Coプロセッサの 1 つ以上は、ツリー検索 Coプロセッサ、チェックサム・コプロセッサ、ストリングコピー・コプロセッサ、エンキュー・コプロセッサ、データストア・コプロセッサ、C A B コプロセッサ、カウンタ・コプロセッサ、及びポリシ・コプロセッサを含むグループから選択される、前記 (19) 記載の方法。

(21) 実行スレッド間の優先順位を確認するため Coプロセッサ実行インタフェース・アービタが用いられる、前記 (20) 記載の方法。

(22) データ・スレッド間の優先順位を確認するため Coプロセッサ・データ・インタフェース・アービタが用いられる、前記 (20) 記載の方法。

(23) 各スレッドと前記 Coプロセッサのうち少なくとも 1 つの間に F I F O バッファを提供するステップを含む、前記 (20) 記載の方法。

(24) 前記 F I F O バッファは各スレッドと前記カウンタ・コプロセッサの間にある、前記 (23) 記載の方法。

(25) 前記 F I F O バッファは各スレッドと前記ポリシ・コプロセッサの間にある、前記 (23) 記載の方法。

(26) 前記 C L P により実行される特定の操作命令を提供するステップを含み、該実行の結果、Coプロセッサ・オペレーションを制御するコマンドが得られ、該コマンドは前記 C L P と Coプロセッサの間のインタフェースを流れる、前記 (18) 記載の方法。

(27) 前記操作命令により特定の Coプロセッサ・オペレーションの条件付き実行が可能になる、前記 (26) 記載の方法。

(28) 前記実行は直接的または間接的である、前記 (27) 記載の方法。

(29) 前記システムが、特定の Coプロセッサ・コマンドに関する予測応答時間に従って、待ち時間の長いイベントと待ち時間の短いイベントを識別し、アクティブなスレッドの実行が待ち時間の長いイベントにより中断したときに他のスレッドにフル制御を与えるか、またはアクティブなスレッドの実行が待ち時間の短いイベントにより中断したときに他のスレッドに一時的制御を与える、命令を提供するステップを含む、前記 (18) 記載の方法。

【図面の簡単な説明】

【図 1】 2 つのコア言語プロセッサと Coプロセッサを持つプロトコル処理ユニットを示す図である。



【図2】2つのコア言語プロセッサとコプロセッサがインタフェースを取るプロトコル処理ユニットを示す図である。

【図3】コア言語プロセッサと選択されたコプロセッサのインタフェースを示す図である。

【図4】コプロセッサ実行インタフェースとコア言語プロセッサを複数のコプロセッサに接続するコプロセッサ・データ・インタフェースを示す図である。

【図5】コプロセッサ実行命令フォーマットを示す図である。

【図6】コプロセッサ実行命令フォーマットを示す図である。

【図7】コプロセッサ実行命令フォーマットを示す図である。

【図8】コプロセッサ実行命令フォーマットを示す図である。

【図9】コプロセッサ特機命令フォーマットを示す図である。

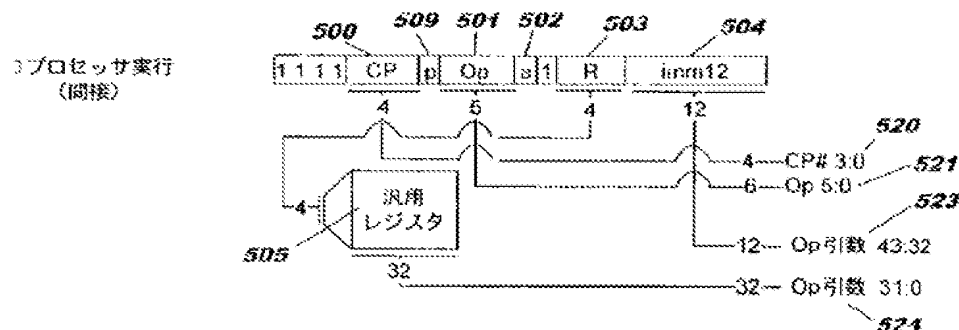
【図10】コプロセッサ待機命令フォーマットを示す図である。

【符号の説明】

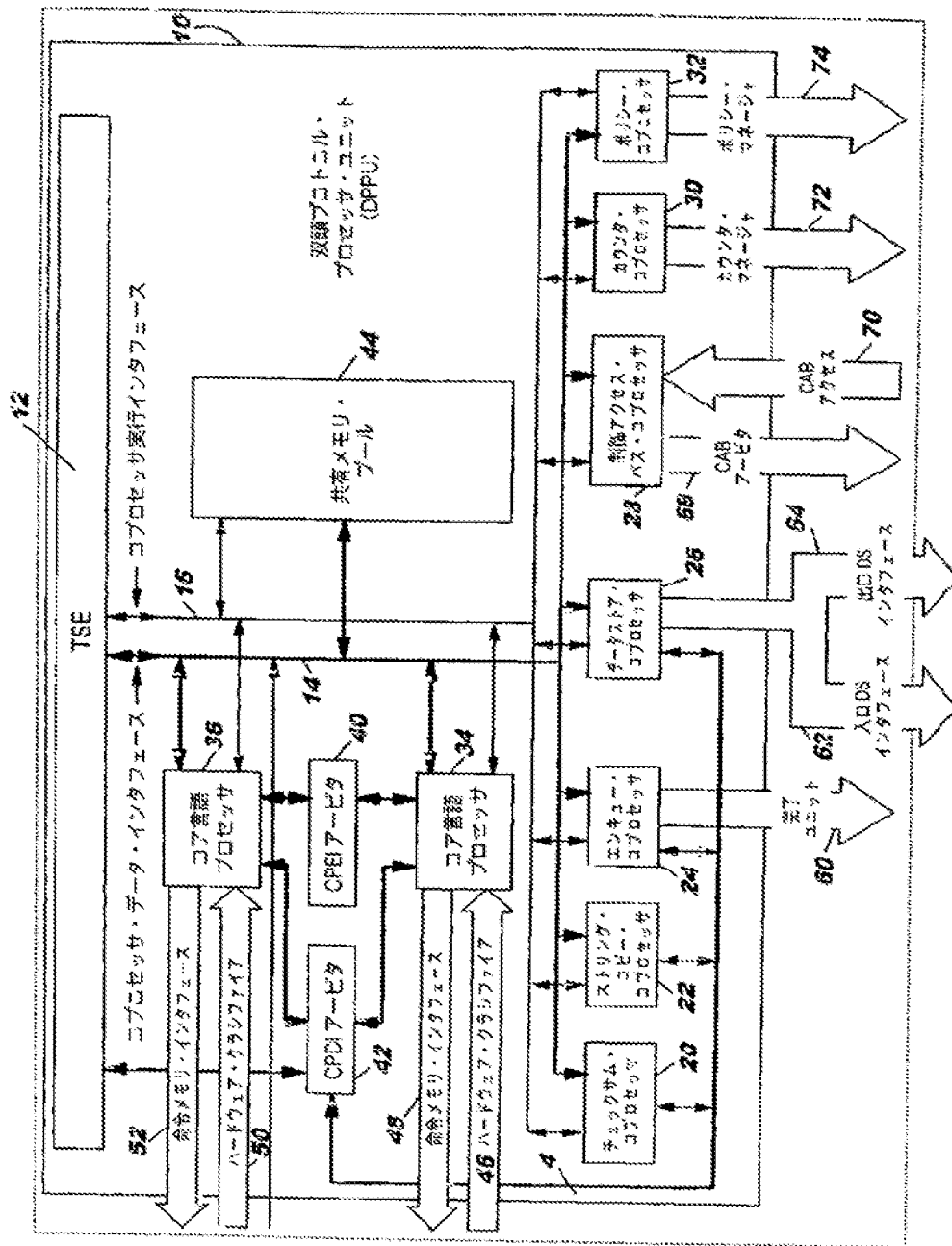
- 10 プロトコル・プロセッサ・ユニット (PPU)
- 12 ツリー検索エンジン
- 14 データ・インタフェース
- 16 実行インタフェース
- 20 チェックサム・コプロセッサ
- 22 スtring・コピー・コプロセッサ
- 24 エンキュアー・コプロセッサ
- 26 データストア・コプロセッサ
- 28 制御アクセス・バス・コプロセッサ
- 30 カウンタ・コプロセッサ
- 32 ボリシ・コプロセッサ

- 34、36 コア言語プロセッサ (CLP)
- 40 コプロセッサ実行インタフェース (CPE) アービタ
- 42 コプロセッサ・データインタフェース (CPDI) アービタ
- 44 共有メモリ・プール
- 56 命令メモリ
- 68 CABアービタ
- 74 インタフェース
- 76、78 FIFOバッファ
- 80 汎用レジスタ
- 82 専用レジスタ
- 84 スカラ・レジスタ
- 86 アレイ・レジスタ
- 88 命令フェッチ／デコード／実行ユニット
- 102 実行ユニット
- 110 ツリー検索メモリ (TSM) アービタ
- 112 フレーム・データ・メモリ
- 401、450 コプロセッサ
- 413 Op引数
- 414 ビジー信号
- 419、420、421、422、423、424 書き込みインタフェース
- 421 コプロセッサ・レジスタ識別子
- 427、428、429、430、431、432、433 読取りインタフェース
- 450 選択コプロセッサ
- 505 32ビット汎用レジスタ
- 520 コプロセッサ識別子
- 523 上位12ビット
- 524 コマンド情報

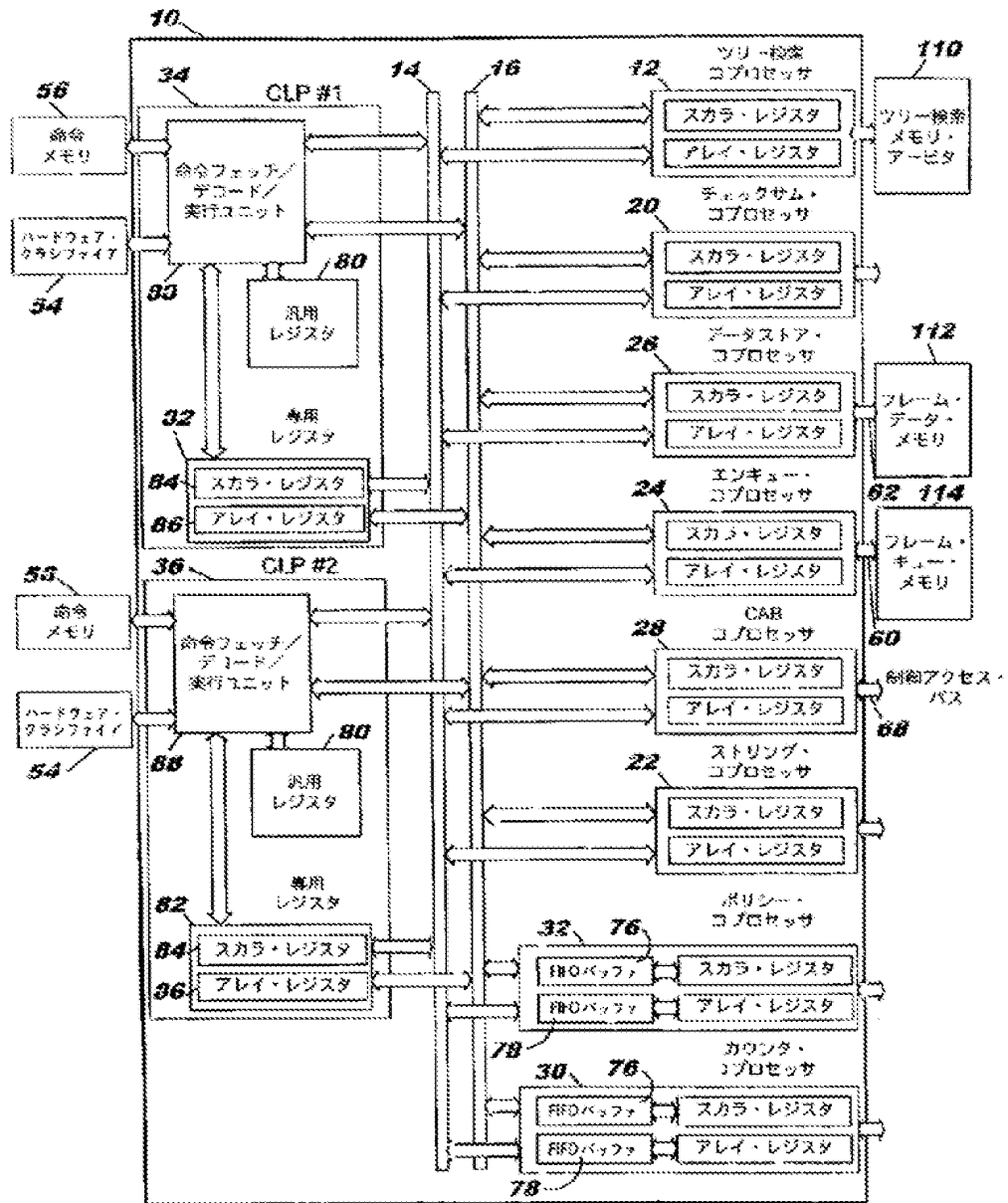
159



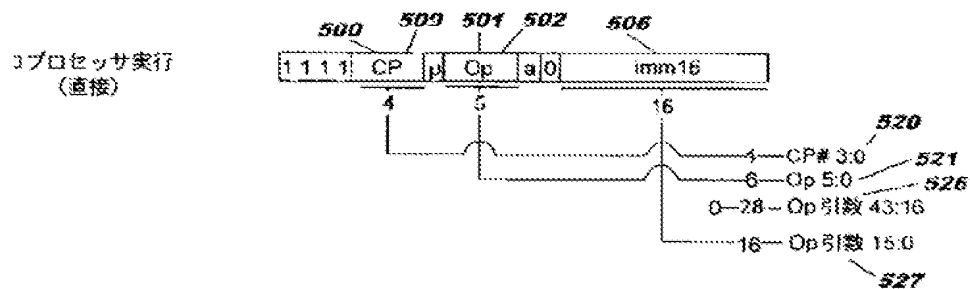
【図1】



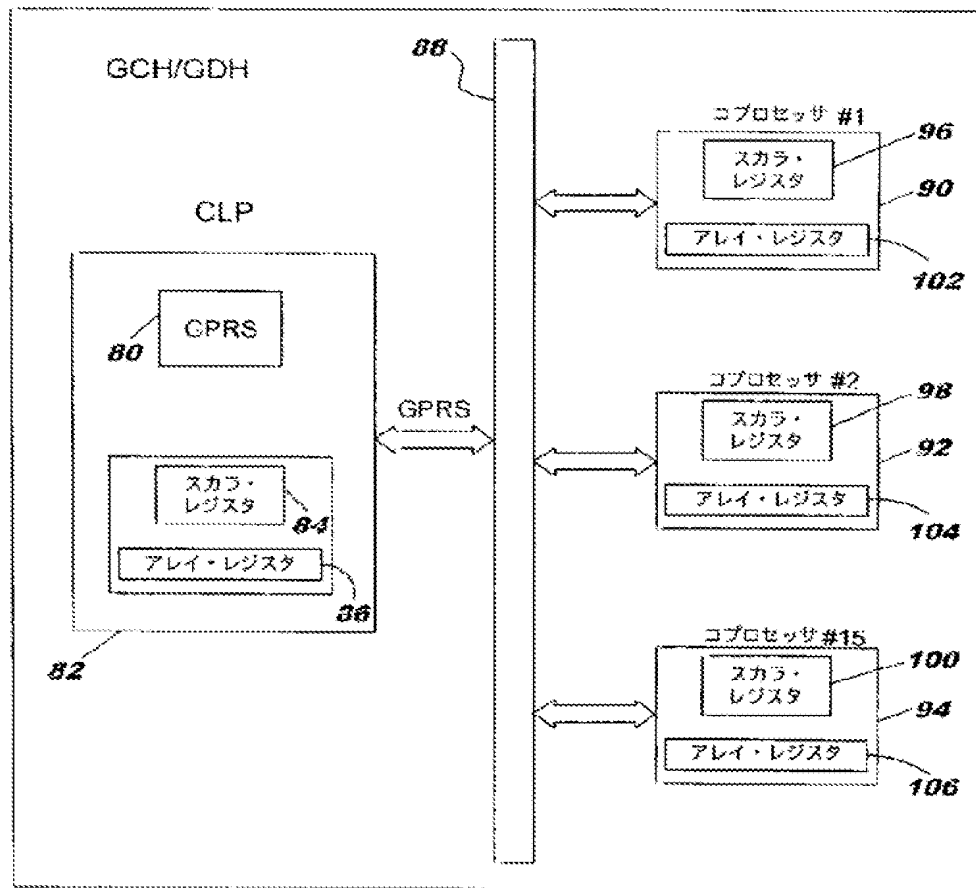
【図2】



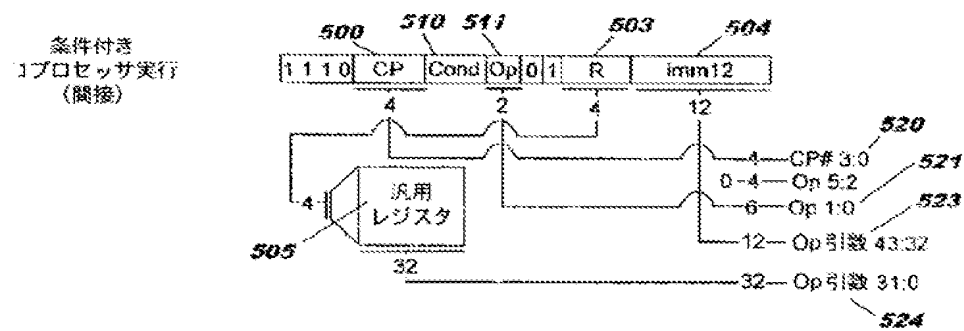
【図7】



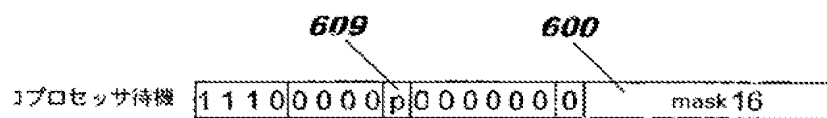
【図3】



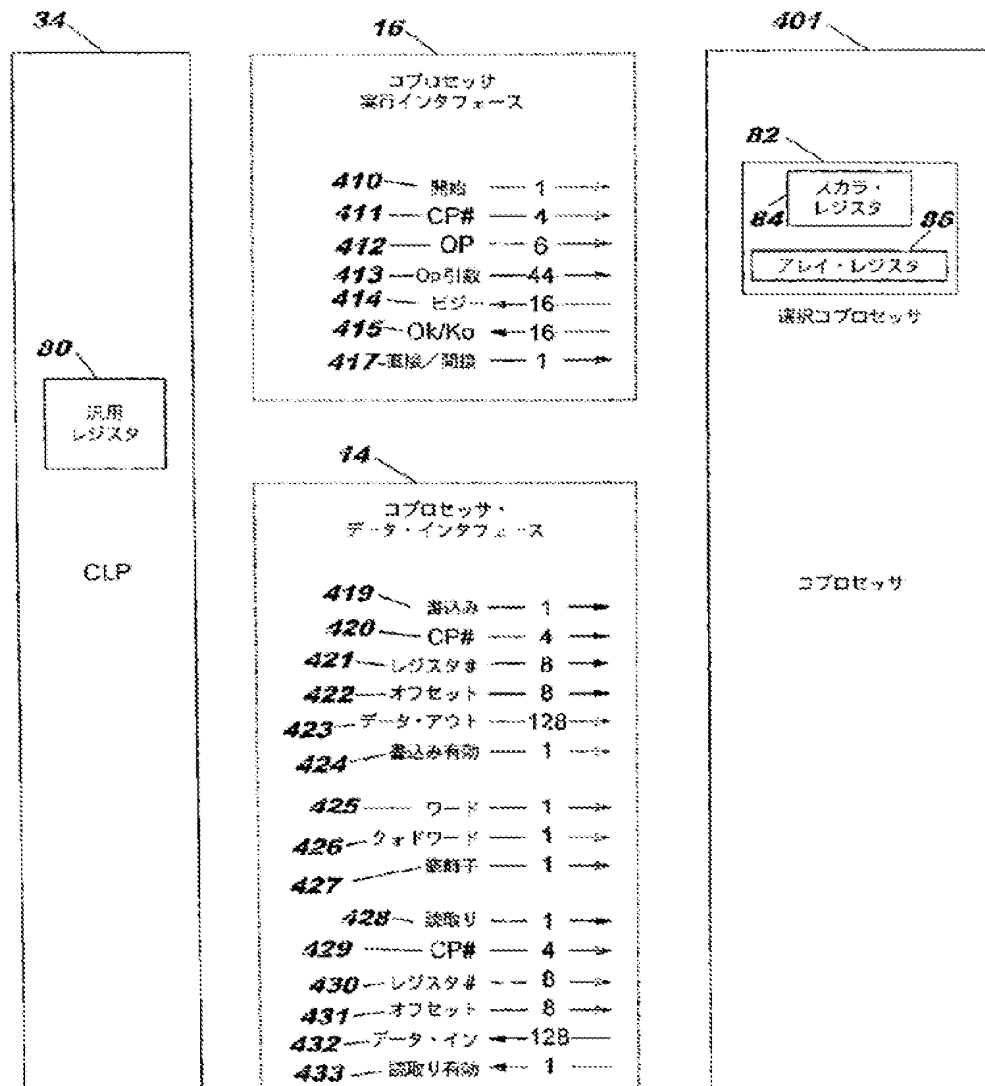
【図6】



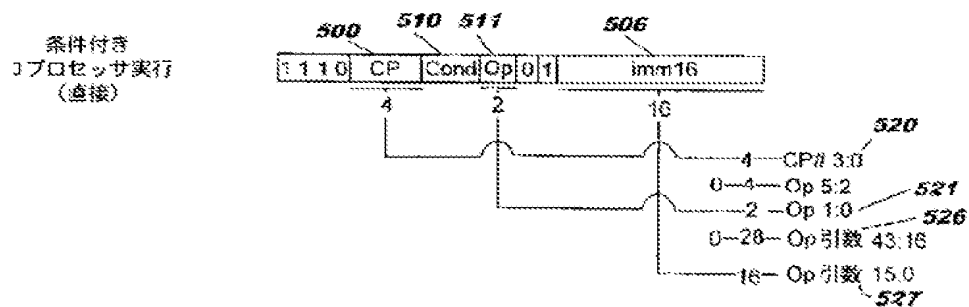
【図9】



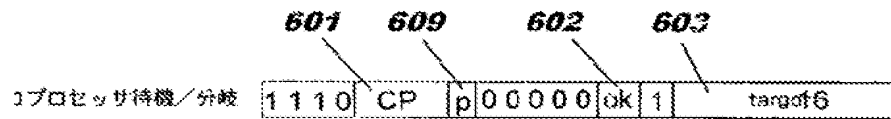
□ □ □



[18]



【図10】



フロントページの続き

(72)発明者 ゴードン・テイラー・デイビス  
アメリカ合衆国27514、ノース・カロライ  
ナ州チャペル・ヒル、フランクリン・リッ  
ジ 97603

(72)発明者 マルコ・シィ・ヘッツ  
アメリカ合衆国27612、ノース・カロライ  
ナ州ローリー、ナンバー 308、グラン  
ド・メナー・コート 4109

(72)発明者 ロス・ボイド・リーベンス  
アメリカ合衆国27511、ノース・カロライ  
ナ州カーリー、ウィランダー・ドライブ  
123

(72)発明者 マーク・アンソニー・リナルディ  
アメリカ合衆国27713、ノース・カロライ  
ナ州ダーハム、クイーンズバリー・サーク  
ル 1201

Fターム(参考) 5B013 DD03  
5B098 AA10 GA04 GA05 GA07 GB09  
GB14 GC03